

```

/* @(#) $Header: /home /pharos /devel /devioc07 /cPCI_noGPIB /cPCIApp /src /RCS /psscFunc.c,v 1.4 2000 /04 /22 00:42:47 susanna E.
*/
/*
*          Functions called by the ALS Power Supply General Subroutine Record
*/
/* Subroutine record inputs:
*      NPP links to other EPICS records —
*      INPA  Setpoint
*      INPB  Time constant (tau)
*      INPC  Ramp Rate (R)
*      INPD  Current Monitor
*      INPK  Power Supply On /Off Control
*
*      Constants —
*      INPE  High Limit
*      INPF  Low Limit
*      INPG  Ramp Rate Limit
*      INPH  Ramper function name
*      INPI  Name of DAC ao record
*      INPJ  Number of bits in DAC
*      INPL  Name of ao record where requested current is set
*/

/* Subroutine record outputs (only for debug monitoring):
*      VALA  Update rate (T)
*      VALB  Maximum step (R*T)
*      VALC  Setpoint after limit check (y1)
*      VALD  Filter coefficient "A": (2*tau-T)/(2*tau+T)
*      VALE  Filter coefficient "B": 1-A
*      VALF  Ramper function address
*      VALG  Ramper struct address
*      VALH  Least meaningful change in current
*/

#include <vxWorks.h>
#include <stdlib.h>
#include <stdio.h>
#include <memLib.h>
#include <string.h>
#include <symLib.h>
#include <sysSymTbl.h> /* for sysSymTbl */
#include <a_out.h> /* for N_TEXT */
#include <math.h>
#include <logLib.h>

#include "dbDefs.h"
#include "dbAccess.h"
#include "genSubRecord.h"
#include "recGbl.h"

#include "pscStruct.h"

extern int PSCRegister(void (*func)(void *), void *parg, char *fname,
                    char *argname);

/* #define DEBUG */

long
pscInit(struct genSubRecord *pgsub)
{
    struct psc_ramp *p;
    DBADDR *pPV = (DBADDR *)NULL;
    double hi, lo, bits, least;
    double lt;

```

psscInit

...pscInit

```

char          ctlPVName[64];
ULONG        *pul;
double       *pdbl;
char         sub_type;
void         *sub_addr;
char         temp[40];
char         *ufunct;
long         status;
STATUS       ret;

/* Allocate psc_ramp struct */
p = (struct psc_ramp *)calloc(1, sizeof(struct psc_ramp));
pgsub->dpvt = p;
/* pointer address to output g (VALG) */
pul = (ULONG *)pgsub->valg;
*pul = (ULONG)p;
#ifdef DEBUG
    printf("PSC: Struct addr = 0x%08lx\n", (ULONG)p);
#endif
/* Note: entire psc_ramp struct is initialized to zero by calloc(). */

/* Allocate and init dbAddr struct for PS Current Output PV */
pPV = (DBADDR *)calloc(1, sizeof(DBADDR));
strcpy(ctlPVName, pgsub->inpi.value.constantStr);
strcat(ctlPVName, ".VAL");
if (dbNameToAddr(ctlPVName, pPV) != 0) {
    printf("PSC ERROR: cannot find PV %s\n", ctlPVName);
}
#ifdef DEBUG
    else {
        printf("PSC: dbNameToAddr(%s) OK.\n", ctlPVName);
    }
#endif
/* Save dbAddr struct's address in psc_ramp struct */
p->ppsc_pv = pPV;

/* Allocate and init dbAddr struct for PS Current Request PV */
pPV = (DBADDR *)calloc(1, sizeof(DBADDR));
strcpy(ctlPVName, pgsub->inpl.value.constantStr);
strcat(ctlPVName, ".VAL");
if (dbNameToAddr(ctlPVName, pPV) != 0) {
    printf("PSC ERROR: cannot find PV %s\n", ctlPVName);
}
#ifdef DEBUG
    else {
        printf("PSC: dbNameToAddr(%s) OK.\n", ctlPVName);
    }
#endif
/* Save dbAddr struct's address in psc_ramp struct */
p->preq_pv = pPV;

/* Clear "alive" and "changed" flags; set "init" flag. */
p->alive = FALSE;
p->changed = FALSE;
p->init = TRUE;

/* Power supply is shut off during reboot; set flags accordingly */
p->switch_on = FALSE;
p->transition = FALSE;
p->send_zeroes = FALSE;
p->got_zero = TRUE;
p->sent_warning = FALSE;

```

...pscInit

```

/* Fill in internal struct pointers */
p->pcurvals = &(p->time_const);
p->pnewvals = &(p->new_time_const);

/* Copy high and low limits to structure */
hi = atof(pgsub->inpe.value.constantStr);
#ifdef DEBUG
printf("PSC: high limit is %f A.\n", hi);
#endif
p->output_max = hi;
lo = atof(pgsub->inpf.value.constantStr);
#ifdef DEBUG
printf("PSC: low limit is %f A.\n", lo);
#endif
p->output_min = lo;

/* Calculate smallest possible change and write to output and structure */
bits = atof(pgsub->inpj.value.constantStr);

/* Calculate resolution slightly greater than hardware capability */
least = (hi - lo) / pow(2.0, bits+2);
#ifdef DEBUG
printf("PSC: smallest change is %e A.\n", least);
#endif
pdbl = (double *)pgsub->valh;
*pdbl = least;
p->smallest_change = least;

/* Get loop time: inverse of sys clock rate. */
lt = 1.0 / sysClkRateGet();
#ifdef DEBUG
printf("PSC: loop time is %f sec.\n", lt);
#endif

/* Copy update rate to structure and to output A */
p->loop_time = lt;
pdbl = (double *)pgsub->vala;
*pdbl = lt;

/* Copy ramp rate limit to structure */
lt = atof(pgsub->inpg.value.constantStr);
#ifdef DEBUG
printf("PSC: ramp rate limit is %f A/sec.\n", lt);
#endif
p->ramp_rate_limit = lt;

/* Read Ramper function name from input H */
ufunct = pgsub->inph.value.constantStr;
#ifdef DEBUG
printf("PSC: Ramper function name is %s.\n", ufunct);
#endif

/* Get function address (code from genSubRecord.c) */
if( ufunct[0] != '\0' )
{
temp[0] = 0;
/* all global variables start with _ */
/* ... except for PowerPC BSP's! */
#ifdef CPU_FAMILY == PPC
#else
if( ufunct[0] != '_' )
strcpy(temp, "_");
#endif
strcat(temp, ufunct);
ret = symFindByName( sysSymTbl, temp, (void *)&sub_addr, (void *)&sub_type );
if( ret < 0 || ((sub_type & N_TEXT) == 0) )
{

```

```

        recGblRecordError(S_db_BadSub,(void *)pgsub,"genSubRecord(init_record)");
        status = S_db_BadSub;
    }
    else
    {
        /* Write Ramper function address to output F. */
        pul = (ULONG *)pgsub->valf;
        *pul = (ULONG)sub_addr;
#ifdef DEBUG
        printf("PSC: Ramper function addr is 0x%08lx.\n", (ULONG)sub_addr);
#endif
    }
    else {
        printf("PSC ERROR: Null Ramper function name.\n");
    }

    /* Register ramper for this PS with the tasker */
    if (PSCRegister((void *)pgsub->sub_addr, p, ufunc, pgsub->name) != 0) {
        printf("PSC ERROR: PSCRegister() failed.\n");
    }
#ifdef DEBUG
    else {
        printf("PSC: PSCRegister Ramper OK.\n");
    }
#endif
    return 0;
}

```

**long**

```

pScProcess(struct genSubRecord *pgsub)
{

```

```

    struct psc_ramp      *p;
    USHORT               *pb;
    double               *pdbl;
    double               *plim;
    double               setp, tau, T, R, KA, KB, RT;
    BOOL                 switch_on;

    p = pgsub->dpvt;
    T = p->loop_time;

    /* If power supply has just been switched on, set flag for ramper */
    /* switch_on size note:
       INPK is an EPICS enum, which is a short in size;
       switch_on is a VxWorks C BOOL, which is an int (long) in size.
    */
    pb = (USHORT *)pgsub->k;
    switch_on = (BOOL)*pb;
    if (switch_on && !p->switch_on) {
        p->transition = TRUE;
        /* Ramper will clear the send_zeroes flag */
        p->send_zeroes = TRUE;
        p->got_zero = FALSE;
        p->sent_warning = FALSE;
#ifdef DEBUG
        printf("PSC: power supply switch change detected.\n");
#endif
    }
    p->switch_on = switch_on;

```

*...pScInit**pScProcess*

```

/* Copy tau into structure */
    pdbl = (double *)pgsub->b;
    tau = *pdbl;
    if (tau < 0.0) tau = 0.0;
    p->new_time_const = tau;
#ifdef DEBUG
    printf("PSC: new tau is %f sec.\n", tau);
#endif

/* Copy R into structure */
    pdbl = (double *)pgsub->c;
    R = *pdbl;
    p->new_ramp_rate = R;
#ifdef DEBUG
    printf("PSC: new R is %f A/sec.\n", R);
#endif

/* Obtain requested set point */
    pdbl = (double *)pgsub->a;
    setp = *pdbl;
#ifdef DEBUG
    printf("PSC: new setpoint is %f A.\n", setp);
#endif

/* Some special cases are handled in the next if ... else construction.
 * 1. The power supply should always be ramped to zero before being
 *    switched ON. This "if" clause enforces that rule.
 *    If the power switch has changed state to ON, the control software
 *    will not allow a request other than zero until it has first seen a
 *    zero request.
 */
    if (p->transition) {
        if (!p->got_zero) {
            if (setp == 0.0) {
                p->got_zero = TRUE;
                p->transition = FALSE;
            }
            else setp = 0.0;
        }
    }

/*
 * 2. The first non-zero request that comes through when the power switch
 *    is OFF will cause a warning message to be sent to the log.
 *    This prevents a flood of log messages. The non-zero request may
 *    be inadvertent or may be someone testing the ramper software.
 */
    else {
        if (!switch_on) {
            if (setp != 0.0 && !p->sent_warning) {
                /* name of setting request record */
                logMsg(
                    "WARNING - nonzero %s while PS switched off.\n",
                    (int) pgsub->inpl.value.constantStr,
                    0,0,0,0,0);
                p->sent_warning = TRUE;
            }
        }
    }

/* Check setpoint against limits and change if necessary */
    /* high limit */
    plim = (double *)pgsub->e;
    if (setp > *plim) setp = *plim;
    else {

```

```

        /* low limit */
        plim = (double *)pgsub->f;
        if (setp < *plim) setp = *plim;
    }
#ifdef DEBUG
    printf("PSC: new limited setpoint is %f A.\n", setp);
#endif

    /* Copy resulting setpoint to output C and to structure */
    pdbl = (double *)pgsub->valc;
    *pdbl = setp;
    p->new_limited_set = setp;

    /* Calculate filter coefficients; put them in struct and outputs D and E. */
    KA = ((2*tau)-T)/((2*tau)+T);
    /*
    KB = T/(2*tau)+T;
    */
    KB = 1 - KA;
    p->new_filt_coef_a = KA;
    p->new_filt_coef_b = KB;
    pdbl = (double *)pgsub->vald;
    *pdbl = KA;
    pdbl = (double *)pgsub->vale;
    *pdbl = KB;
#ifdef DEBUG
    printf("PSC: Filter coefficient val's are (A) %f, (B) %f\n", KA, KB);
#endif

    /* Calculate RT; write to output B and structure. */
    RT = R*T;
    p->new_max_step = RT;
    pdbl = (double *)pgsub->valb;
    *pdbl = RT;
#ifdef DEBUG
    printf("PSC: New RT is %f A.\n", RT);
#endif

    /* Set structure changed flag. */
    p->changed = TRUE;

    /* If first time through, read current into struct's previous settings... */
    if (p->init) {
        p->init = FALSE;
        pdbl = (double *)pgsub->d;
        setp = *pdbl;
        p->limited_setpt = setp;
        p->filter_output = setp;
#ifdef DEBUG
        printf("PSC: Current monitor says %f A.\n", setp);
#endif
    }
    /* ...and tell Ramper to start running. */
    p->alive = TRUE;
}

return 0;
}

void
pscRampStructPrint(struct psc_ramp *p)
{
    printf(
        "pcurvals: 0x%08lx          pnewvals: 0x%08lx\n\
        ppsc_pv: 0x%08lx          preq_pv: 0x%08lx\n\
    );
}

```

pscRampStructPrint

*...pscRampStructPrint*

```

alive: %s          changed: %s          init: %s\n\
switch_on: %s     transition: %s      send_zeroes: %s\n\
got_zero: %s     sent_warning: %s\n\
output_max: %f A  output_min: %f A      smallest_change: %e A\n\
loop_time: %f sec time_const: %f sec\n\
ramp_rate_limit: %f A/sec ramp_rate: %f A/sec\n\
filt_coef_a: %e   filt_coef_b: %e       max_step_size: %e A\n\
limited_setpt: %e A filter_output: %e A\n\
prev_lim_setpt: %e A prevflt_out: %e A\n\
new_time_const: %f sec new_ramp_rate: %f A/sec\n\
new_filt_coef_a: %e   new_filt_coef_b: %e\n\
new_max_step: %e A   new_limited_set: %e A\n",
    (ULONG)p->pcurvals, (ULONG)p->pnewvals,
    (ULONG)p->ppsc_pv, (ULONG)p->preq_pv,
    p->alive?"TRUE":"FALSE", p->changed?"TRUE":"FALSE",
    p->init?"TRUE":"FALSE",
    p->switch_on?"TRUE":"FALSE", p->transition?"TRUE":"FALSE",
    p->send_zeroes?"TRUE":"FALSE", p->got_zero?"TRUE":"FALSE",
    p->sent_warning?"TRUE":"FALSE",
    p->output_max, p->output_min, p->smallest_change,
    p->loop_time, p->time_const,
    p->ramp_rate_limit, p->ramp_rate,
    p->filt_coef_a, p->filt_coef_b, p->max_step_size,
    p->limited_setpt, p->filter_output,
    p->prev_lim_setpt, p->prevflt_out,
    p->new_time_const, p->new_ramp_rate,
    p->new_filt_coef_a, p->new_filt_coef_b,
    p->new_max_step, p->new_limited_set);
}

```